PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 3/033, 9/44	A1	(11) International Publication Number: WO 98/37483 (43) International Publication Date: 27 August 1998 (27.08.98)

(21) International Application Number: PCT/CA98/00125

(22) International Filing Date: 19 February 1998 (19.02.98)

(30) Priority Data:
2,197,953 19 February 1997 (19.02.97) CA

(71) Applicant *(for all designated States except US)*: GALLIUM SOFTWARE INC. [CA/CA]; Suite 4000, 303 Moodie Drive, Nepean, Ontario K2H 9R4 (CA).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: JANSSEN, Steve [CA/CA]; (CA). GOTTSTEIN-PURI, Annette [CA/CA]; Gallium Software Inc., Suite 4000, 303 Moodie Drive, Nepean, Ontario K2H 9R4 (CA).

(74) Agents: O'NEILL, T., Gary et al.; Gowling, Strathy & Henderson, Suite 2600, 160 Elgin Street, Ottawa, Ontario K1P 1C3 (CA).

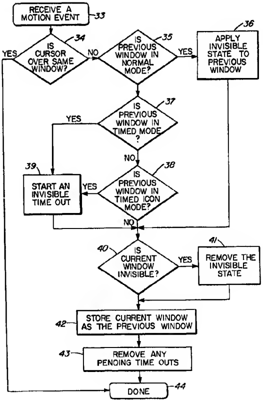
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: USER INTERFACE AND METHOD FOR MAXIMIZING THE INFORMATION PRESENTED ON A SCREEN

(57) Abstract

A system provides a user interface for maximizing an amount of information presented on a computer generated display. The system includes a user interface management system (UMIS) that works with an operating system that manages the computer resources including a display screen and a user input device for tracking position. The UMIS is responsive to the user input device to effect changes in the computer generated display. The UMIS communicates with an application program using the user interface. The user interface has at least a first layer for displaying a primary dynamic image and a second layer for displaying a secondary dynamic image. The second layer has a plurality of display modes including a first mode corresponding to at least a portion of the second layer assuming an invisible state in the computer generated display. Selection of the display modes is effected by signals received from the user input device. Both the first and second layers are updated regardless of the mode of the second layer. The user interface is particularly applicable to air traffic control where the first layer is used for displaying a map plotting aircraft locations and the second layer is used for displaying various data concerning the aircraft, both being updated continuously.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

User Interface and Method for Maximizing the Information Presented on a Screen

The present invention generally relates to methods for displaying graphical information on a computer system, and more particularly, the present invention relates to a computer controlled display system for organizing the display of a high volume of information and a user interface to allow the operator to readily view all sets of information.

Background of the Invention

A windowing environment is system software that manages interactions between a user and an application program executing on a computer through a graphical display monitor. Typically, the graphical display is arranged to resemble an electronic "desktop", and each sheet of information on the "desktop" is displayed in a rectangular region of the screen called a "window". The windows on the "desktop" can be organized in a variety of different ways. They can be tiled so the contents of each window are totally visible to the operator, they can be overlapped so that the contents of a window partially overlays another window, or they can be stacked so that one window completely overlays another window. The windows on the desktop can be used to contain any object, including simple objects such as menus, forms and tables and complex objects such as spreadsheets and radar displays for air traffic control.

In a typical window-based Graphical User Interface (GUI) system (such as Microsoft Windows® or OSF Motif®), a variety of techniques are provided to the operator to manage the windows on the display. The windows can be made larger or smaller, they can be expanded to be the full screen size, they can be moved to a different position on the screen, or they can be reduced to an icon. An icon is a small, visually distinct display object which represents the window.

In a traditional office application, the windows are used to represent static information such as documents and spreadsheets. The content of a window changes only when a change is effected by the operator. However, there is a class of applications where the information in the windows changes dynamically independently of operator intervention. For example, in an Air Traffic Control display, one window may contain a geographic view of the airspace in which aircraft are plotted on the display according to their current position based on radar reports. Another window may have a dynamically changing table summarizing details about each aircraft including information such as current speed and altitude, which is updated based on radar reports.

One of the problems associated with window based graphical user interfaces is the necessity of carefully managing the display screen space. In many complex applications there is a great deal of information which must be displayed to the operator in multiple windows. However, with multiple windows, there is often not enough screen space (screen "real estate") to concurrently view all the important information displayed in various windows. For example, in Air Traffic Control, the focus of the radar operator is on the main situation display window where the operator is tracking the movement of aircraft through the radar plots displayed in the window. The operator also needs to regularly be able to view additional information about the aircraft, weather conditions, etc., as displayed in other windows. However, at the same time, the operator needs to maintain full awareness of the main situation window which reflects the position of all aircraft. The ability to maintain a great number of windows on the screen without obscuring the main window of interest is desired. In these situations, it is too cumbersome and time consuming to use standard window manipulation techniques such as resizing or moving windows.

Other attempts to solve this problem usually involve one of two techniques. The first is to provide either a larger screen surface or multiple screen surfaces to provide enough space to display all of the required data. The drawbacks of this solution are that the additional screen space require more expense and the necessary room to accommodate larger or multiple screens may not be available. The other technique is to dedicate areas of the screen at which to place menus and tables. This reduces the amount of screen space for radar data and for other menus and/or tables which may be required and the operator no longer has the flexibility of deciding for himself the best mix of data for the situation at hand.

A still further problem is that the windowing systems on most modern computers (for X11 windows in the Unix environment) do not update data that lie directly beneath a window. Even if such a window is transparent, it will not be updated. Modern window systems typically cannot process input events on objects that are not drawn.

As will be disclosed, the present invention provides a method of designating windows as invisible so that information in background windows is not obscured, a user interface for viewing and hiding the data in the foreground window on demand, and a method for managing and rendering the displays when working with invisible windows.

Summary of the Invention

One aspect of the present invention seeks to provide the operator with a rapid means of exposing/hiding information in windows. Another aspect of the invention seeks to provide a method and system for updating images which reside beneath a window.

With regard to the first aspect, the present invention provides a method and user interface technique that allows the operator to maintain a large number of windows all containing information necessary for the operator to perform his task, while at the same time not obscuring other windows which are essential to perform the task. This approach significantly increases operator productivity and also increases safety when employed in safety critical applications since it permits the operator to maintain maximum awareness of the main safety critical situation window, while still providing immediate access to the other information necessary for the operator to perform his task.

The invention operates in a standard environment of computer workstation with a graphical display. Information is displayed in "windows" on the graphical display, and the operator interacts with the display with standard input devices such as a keyboard and a mouse. This invention may be embodied in an application program that executes on the workstation or any other type of program, including the Operating System which controls the workstation.

This invention consists of a user interface which provides the operator with a rapid means to expose and hide information in invisible windows. When the information in windows is hidden, the "invisible" windows can be totally invisible (i.e., there is no visual indication of their location), the windows may have a title bar that is visible, the windows may have a window border that is visible, or the windows may have a title bar and window border that is visible. These latter states provide the operator with a visual clue as to the location of the hidden window. In all these cases, the contents of the invisible window are not displayed and the background window is fully visible through the invisible window.

The user is provided the ability to designate each invisible window as "normal", "timed", "locked", or "timed icon". The user is also able to reduce an invisible window to an icon at any time. When an invisible window is reduced to an icon no window operations can be performed on the window until the icon is raised back into an invisible window.

When in "normal" mode, the contents of the window are exposed when the cursor moves into the area of the window. The window contents can be exposed either by allowing the window to be displayed on an opaque background, which enhances legibility of the window contents, or on a transparent background, which enables the contents of

the background windows to be visible underneath the invisible window. The window contents are hidden again by simply moving the cursor away from the window.

In "timed" mode, the contents of the window are exposed in the manner described above for a specified period of time, at which time the window automatically returns to its invisible state. In the "locked" mode, the contents of the window are exposed in the opaque manner described above until another mode is selected for the window. In the "timed icon" mode, the contents of the window are exposed in the manner described above for a specified period of time, at which time the window is automatically reduced to an icon.

With regard to the second aspect, the invention can render windows which are not directly contained in the computer's native windows and provides unique drawing strategies to ensure that updates occur to data that lay beneath a given window.

Thus, windows can occupy the same display area, yet the operator can rapidly select which objects to view without losing situational awareness. The expense of larger screen surface area and the restriction of dedicated table/menu areas is eliminated.

Brief Description of the Drawings

These and other features of the invention will become more apparent from the following description in which reference is made to the appended drawings in which:

Figure 1 is a pictorial front view of a typical computer;

Figure 2 is a display presentation of an opaque window technique;

Figure 3 is the display presentation of **Figure 2,** using a transparent window technique according to the invention;

Figure 4, illustrates the display of **Figure 2** when the cursor is moved over the transparent window;

Figure 5 shows the display mode menu;

Figure 6 shows the general software architecture of a transparent window system according to the invention;

Figure 7 is a flowchart illustrating the logic used to process a motion event detected on a window;

Figure 8 is a flowchart illustrating the logic used to process a transparent time out event;

Figure 9 is a flowchart illustrating the logic used to process a timed icon time out event;

Figure 10 shows application oriented objects contained in the system of Figure 6;

Figure 11 is a flowchart illustrating the logic used to update areas located under a transparent window;

Figure 12 is a flowchart illustrating the logic used to implement behavior for objects which are invisible.

Some numeral references will denote same parts throughout the description.

Detailed Description of Preferred Embodiments of the Invention

Figure 1 depicts a typical computer system suitable for supporting the present invention. **Figure 1** is but one of many computer configurations that can support the present invention. **Figure 1** shows a computer system consisting of an output device **1** (display monitor), input devices (mouse **5** and keyboard **6**) and a system chassis **3**. Other configurations might include flat screen displays, X-terminals, trackballs, touch screens and other devices.

The system chassis **3** contains components such as the Central Processing Unit (CPU), memory, hard disk storage, power supply, local area network , parallel printer I/O, serial I/O, and video generation hardware. The system chassis **3** may contain optional items such as CDROMs, tape drives, sound generation facilities and others.

The present invention provides a means to efficiently manage screen space when there are conflicting demands of high priority information in background windows which occupy large portions of the display surface **2** and numerous information windows overlaying it. The present invention can be implemented at four different levels in a computer system. It can be applied in the graphics generation hardware, the operating system graphics subsystem software (X Windows, OpenGL ®, Microsoft Windows ®, Apple MacO/S ®, etc.), a User Interface Management System (UIMS), or in application software. The discussion below deals mostly with implementation of the present invention in a UIMS and associated application programs. A UIMS is a layer of software which manages all display and user device input activities for a specific application. The description is divided into three parts: Part 1 describes the invention as manipulated by the operator; Part 2 describes the invention in terms of the software algorithms which describe its operation; Part 3 describes the features of the UIMS to support the event capture and drawing techniques necessary.

PART 1

Figure 2 depicts a typical application display presentation where the present invention would greatly enhance operator usability and operator access to important

data. In this view, normal windowing techniques are in force. The high importance background data **8** (radar return information in this case) is being obscured by multiple overlapping user interface windows **7**.

These overlapping windows **7** are a necessary part of the user's displayed information set. They are explicitly requested by the user and necessarily appear on top of the background data **8**. The present invention provides the operator with techniques to control their property of covering underlying data.

Figure 3 depicts the same application with the invisible window feature operating. Notice now that the overlapping windows **10** no longer obscure the background radar data **9**. The overlapping windows' title bars and window frames **11** are still visible giving the user visible cues as to where the windows will be when opaque. The view of the background radar in **Figure 3** is vastly improved over that in **Figure 2**.

The technique which the operator uses to toggle between opaque and invisible is as simple as moving the cursor **12** over the invisible window which the operator wishes to view. In **Figure 3** the cursor **12** is over the background data 9, not an overlapping window **10**. This causes all overlapping windows **10** to be invisible.

Figure 4 demonstrates the system's reaction to moving the cursor over an invisible window. Notice that the cursor **16** is now over the table titled "Category Select - CATEGORY" **15**. This table is now opaque while the others **14** remain invisible. The background data **13** is covered by tables **15** but remains unaffected by tables **14**.

Further control of the invisible effect is afforded the operator through a button in the title bar of the tables **17**. Activating this button causes the display mode menu **18** to be activated. **Figure 5** depicts the display mode menu. The menu consists of four buttons **19, 20, 21, 22**. Each button has a graphical icon and textual label. When a button is selected, its icon is placed in the display mode menu **18** as seen in buttons **17** of **Figure 4**. Each window can be placed in a separate mode.

Button **19** is used to activate the *Normal* display mode. In this mode all overlapping windows are invisible by default. A window becomes opaque when the cursor is placed within its extent. When the cursor leaves its extent the window becomes invisible again.

Button **20** is used to activate the *Timed* display mode. In this mode all overlapping windows are invisible by default. A window becomes opaque when the cursor is placed within its extent. When the cursor leaves its extent the window becomes invisible only after a time-out has occurred. This time-out is programmable but is

typically 10 seconds. The time-out is terminated if the cursor is placed over the window before it turns invisible.

Button **21** is used to activate the *Locked* display mode. In this mode the overlapping windows are always opaque.

Button **22** is used to activate the *Timed-Icon* display mode. In this mode all overlapping windows are invisible by default. A window becomes opaque when the cursor is placed within its extent. When the cursor leaves its extent a two step time-out process begins. After the first time-out, the window goes invisible and the second time-out begins. After the second time-out, the window automatically iconifies. Window **23** in **Figure 4** illustrates an iconified window. These time-outs are configurable but are typically ten seconds. The time-outs are terminated if the cursor is placed over the window before the operation completes.

The activation of any of the *Normal, Locked, Timed,* or *Timed-Icon* buttons causes the mode data field on the associated window to be updated to the appropriate mode.

PART 2

Figure 6 illustrates the general software architecture of the invisible window system. A window has three pieces of data associated with it for the purposes of this discussion. Each window has a unique identity stored in window id. Each window can operate in one of four modes: *Normal, Locked, Timed,* or *Timed-Icon*. Finally, each window can be in one of three states: *invisible, opaque* or *iconified.*

The following steps describe the action of moving the mouse from an opaque state window onto the background data.

The user moves the mouse **26** pointer from an opaque state window to the background which generates a hardware event **32**. The operating system **25** services the hardware mouse event **32** and passes a message **29** to the UIMS **24**. The UIMS **24** maintains the extent of all objects it manages. It traverses this list examining each window's extent to determine if the cursor now lies within the same window as the previous event. If the UIMS **24** does not find a match, the UIMS checks if the cursor was previously on an opaque window. If true, the UIMS **24** next determines if that window has been designated as a potentially invisible window. If the window is designated as potentially invisible the application defined *invisible state* **30** is applied to the window by the UIMS **24**. The user may specify the *invisible state* **30** to the UIMS such that the entire window becomes invisible or just user specified portions of it become invisible.

The application of this state causes the UIMS **24** to redraw the window on the display **28** according to the *invisible state* specifications.

The following steps describe the action of moving the mouse from the background data onto a window which is currently invisible.

The user moves the mouse **26** pointer over an invisible window which generates a hardware event **32**. The operating system **25** services the hardware mouse event **32** and passes a message **29** to the UIMS **24**. The UIMS **24** maintains the extent of all objects it manages. It traverses this list examining each window's extent to determine if the cursor now lies within that extent. If the UIMS **24** finds a match, it next determines if that window has been designated as an invisible window. If the window is designated as potentially opaque the application defined *normal state* **30** is applied from the window by the UIMS **24**. The application of this state causes the UIMS **24** to redraw the window on the display **28** according to the *normal state* specifications. The UIMS **24** notes the fact that the cursor is over a particular window which is now opaque.

The details of how the UIMS searches input matches on windows and how data beneath invisible windows is updated are detailed in PART 3.

Figures 7 through **10** depict flow chart views of the logic within the UIMS **24**. When the lower levels of the UIMS (see Part 3) detect a motion event on a window, an algorithm depicted in **Figure 7** is initiated **33**. The algorithm compares **34** the window id in the event message with the stored previous window id. If the window ids are the same, the algorithm terminates **44**.

If the current and previous window ids do not match, the mode of the window is tested **35**. If the window is in *normal* mode then apply the invisible state to the window **36**. If the window is not in *normal* state then test to see if it is in *Timed* mode **37**. If the window is in *Timed* mode then schedule a "timed invisible" time out event to start **39**. If the window is not in *Timed* mode then test to see if it is in *Timed-Icon* mode **38**. If the window is in *Timed-Icon* mode then schedule a "timed invisible" time out event to start **39**. Regardless of which mode the current window is in the state of the previous window is now tested **40**. If the current window is invisible, remove the invisible state **41** to make the window opaque. Next, store the current window id as the previous window id **42** and remove any time outs that may be pending on the window from previous cursor movement **43**. This algorithm now terminates and regular processing continues.

Figure 8 depicts the algorithm used to process "timed invisible" time out events. The algorithm is initiated when a time out of type "timed invisible" is received **45**. The window state is set to invisible and the window is checked to see if it is in *Timed-Icon*

mode **48**. If it is a new time out of type "timed icon" is initiated **47**. The algorithm now terminates **49**.

Figure 9 depicts the algorithm used to process "timed icon" time out events. The algorithm is initiated when a time out of type "timed icon" is received **50**. The window state is set to iconified **51**. The algorithm now terminates **52**.

PART 3

To support the above described behavior the invention embodies the capabilities described below.

The UIMS of **Figure 6** contains *application-oriented objects*, hierarchical InterMAPhics *display lists*, and one or more *rasterized images*. Representations of the objects are maintained in the display lists by the *presentation manager*, and the rasterized images are created from the display lists by the *renderer*. This is shown in detail in **Figure 10.**

The application-oriented objects include the user interface windows used to present information to the operators. Each object has the following properties:

- a number of representations each defined by an overall display layer and position, individually layered and positioned object components, and symbolic references to UIMS graphical constructs for each component;
- a number of logical states defined by UIMS keys linked to representation changes; and
- a number of behaviors defined by UIMS actions linked to symbolic object events.

The specification of object properties is via UIMS configuration data, typically loaded at system startup.

The *presentation manager* maintains the hierarchical display lists according to the current properties of the application-oriented objects. The presentation manager provides the following functionality:

- inserts, modifies, and deletes graphical data in the display lists to reflect the representation properties of the application-oriented objects
- maintains a hierarchical ordering among and within the display lists to reflect the layering properties of the application-oriented objects
- maintains a mapping from all graphical data in the display lists to the application-oriented object with which the data is associated
- translates operating system input events to symbolic object events on an object-by-object basis
- invokes UIMS actions as required to implement the behavior of the objects.

Updates to application-oriented objects and operating system inputs invoke this functionality. Operating system inputs are translated to symbolic object events via UIMS configuration data, typically loaded at system startup.

The hierarchical UIMS *display lists* contain graphical data describing the appearance of the display. The display lists have the following characteristics:

- graphical data consisting of vector-based graphical primitives and symbolic graphical attributes such as color, drawing style, and text characteristics;
- a hierarchical structure, whereby any display list may contain other display lists as well as graphical data;
- the display lists, their contained display lists, and the graphical data are sorted from lowest layer (underlying) to highest layer (overlying) relative to the containing display list.

The presentation manager establishes and maintains these characteristics, which are referenced by the renderer.

The *renderer* maintains rasterized images according to the contents of the display lists. The renderer provides the following functionality:

- traverses the display list hierarchy in either ascending and descending sort order performing vector-by-vector conversion of graphical primitives to raster images (rendering). This traversal can be constrained by an image extent or distance from a specific image pixel;
- determines a set of rendered pixels for each primitive. This is based on the primitive vectors and a translation of the symbolic graphic attributes for drawing styles and/or text characteristics;
- determines a value to be applied against the raster image for each rendered pixel. This is based on a translation of the symbolic graphic attributes for color
- determines a logical function for each rendered pixel specifying how the pixel value is to be applied against the raster image. This is based on a translation of the symbolic attributes for color, drawing styles, and/or text characteristics. For transparent pixels, the logical function leaves the corresponding pixel in the raster image unchanged, while for opaque pixels the logical function replaces the corresponding pixel in the raster image. Other logical functions provide different visual effects;
- updates one or more raster images from the rendered pixel set, values, and functions determined from a primitive;
- copies some or all of one or more raster images to the hardware video memory.

The presentation manager invokes this functionality to request visual updates or to determine which primitives in the display list contain specific pixels in their rendered pixel set. Symbolic graphical attribute translation to pixel sets, values, and logical functions is through a mapping specified by UIMS configuration data, typically loaded at system startup. This mapping supports transparency through transparent colors and hollow fill drawing styles.

The *raster images* are either mapped directly to video memory by the graphics display hardware, or are exact images which can be block copied directly to the video memory. The raster images have the following characteristics:

- an array of pixel values arranged exactly as required for video display;
- overlying objects have been rendered after underlying objects, so that pixels rendered for the overlying objects modify or replace those of underlying objects.

The renderer establishes and maintains these characteristics.

Figure 11 depicts the algorithm used to update areas beneath the transparent user windows. An application-oriented object behind a transparent window is updated through a change in logical state **53**. The presentation manager updates the object representation in the display list by modifying the graphical primitives and attributes associated with that object to reflect its new state **54**. The renderer then traverses the hierarchical display list structure as follows. The current display list is set to the top level display list **55**. While the end of the current display list has not been reached **56** , the next item in the display list is processed **57**. If this item is not an embedded display list **58**, the renderer determines the rendered pixel set, pixel values, and logical functions for the graphical data associated with the item **59**. The renderer updates the raster images from the rendered pixels **60**. This continues until the end of the current display list has been reached **56**. If an item in the display list is itself an embedded display list **58**, the renderer pushes the current display list onto a LIFO stack **61**, and sets the current display list to the embedded display list **62**. The renderer continues processing with this display list **56**. When the current display list has been completely processed **56**, and it is not the top level display list **63**, the renderer sets the most recently pushed display list as the current display list **64**, and continues the processing of that display list **56**. If the completed display list is the top level display list **63**, rendering is complete. The renderer copies the updated raster images to video memory **65**. Note that all graphical items in the display list are rendered **59, 60**, regardless if they are overlain by an opaque or transparent object. In the opaque case, the rendered pixels of the underlying object are

replaced by the rendered pixels of the overlying object **60**. In the transparent case, the rendered pixels of the underlying item are unaffected by the rendered pixels of the overlying item **60**.

Figure 12 depicts the algorithm used to implement behavior for objects which are invisible. The operator moves a tracking device connected to the system hardware. The operating system sends a notification of the motion **66** and a report on the new tracking position to the presentation manager **67**. The presentation manager invokes the renderer to determine which primitives in the display lists, searched from uppermost to lowermost layers, contain the current tracking pixel in their rendered pixel set **68**. Note that the rendered pixel set also contains all transparent pixels for any primitive. If no primitives are found by the renderer **69**, the presentation manager awaits the next operating system event **66**. If the list contains at least one primitive, the presentation manager accesses the application-oriented object associated with the first primitive in the list **70**. If it can map the operating system event to a symbolic object event for the application-oriented object **71**, it invokes the UIMS actions linked to the object event through its behavior definition **72**. The object behavior definition either terminates the input event processing or instructs the presentation manager to propagate it through to the next underlying object **73**. If processing is complete, the presentation manager awaits the next operating system event **66**. If the event is to be propagated **73**, or the event cannot be mapped to a symbolic object event **71**, the presentation manager deletes the primitive from the list **74**, and continues processing while there are primitives remaining in the list **69**.

We Claim:

1. A graphical user interface, comprising:
a first display layer for presenting a primary dynamic image;
a display position indicator responsive to a user input device; and
a second display layer for presenting a secondary dynamic image related to the primary dynamic image and positionable relative thereto, the second display layer having visible or invisible modes and assuming a visible mode in dependence upon a predetermined user selection and the primary and secondary dynamic images being updated regardless of position or mode of the second display layer.

2. A graphic user interface as claimed in claim **1**, wherein the mode of the second display layer is affected when the display position indicator is coincident therewith.

3. A graphic user interface as claimed in claim **1**, wherein the user selection is one of a normally invisible mode, the timed mode, a locked mode or the timed-icon mode wherein for the normally visible mode the second display layer is normally invisible, for the timed mode the second display layer is visible for a predetermined time, then becomes invisible, for the locked mode, second display layer is locked in a visible mode and for the time-icon mode the second display layer is visible for a predetermined time then changes to an icon.

4. A graphical user interface as claimed in claim **3**, wherein the user selection method is by button depression using the user input device.

5. A graphical user interface as claimed in claim **4** wherein the buttons are labeled normal, timed, locked and timed-icon.

6. A system for providing a user interface for maximizing an amount of information presented on a computer generated display, said system comprising:

a user interface management system for receiving messages, from an operating system that manages the computer resources including a display screen and a user input device for tracking position, and in particular, messages in response to the user input device and for sending messages to the operating system effecting changes in the computer generated display and for receiving messages from and sending messages to an application program using the user interface;

the user interface having at least a first layer for displaying a primary dynamic image and a second layer for displaying a secondary dynamic image, the second layer having a plurality of display modes including a first mode corresponding to at least a portion of the second layer assuming an invisible state in the computer generated display, selection of the display modes being in dependence upon signals received from the user input device.

7. A system as claimed in claim **6**, wherein the mode of the second display layer is affected when the display position indicator is coincident therewith.

8. A system as claimed in claim **6** wherein the user selection is one of a normally invisible mode, the timed mode, a locked mode or the timed-icon mode wherein for the normally visible mode the second display layer is normally invisible, for the timed mode the second display layer is visible for a predetermined time, then becomes invisible, for the locked mode, second display layer is locked in a visible mode and for the time-icon mode the second display layer is visible for a predetermined time then changes to an icon.

9. A system as claimed in claim **8**, wherein the user selection method is by button depression using the user input device.

10. A system as claimed in claim **9**, wherein the buttons are labeled normal, timed, locked and timed-icon.

11. A graphical user interface for air traffic control comprising:
a first display layer for presenting an aircraft tracking display;
a second display layer for presenting auxiliary data related to the first display layer;
a display position indicator responsive to the user interface device; and
the second display layer having a plurality of display modes including a mode in which a portion of the second display layer is in an invisible state.

12. A graphical user interface as claimed in claim **11**, wherein the plurality of display modes includes a thermally invisible mode in which the portion of the second display layer is in an invisible state until the display position indicator is coincident with the second display layer.

13. A graphical user interface as claimed in claim **11**, wherein the plurality display modes includes a timed mode in which the portion of the second display layer is in an invisible state until the display position indicator is coincident with the second display layer which then remains in the invisible state for a predetermined time after the display position indicator is no longer coincident therewith, then the portion of the second display layer returns to an invisible state.

14. A graphical user interface as claimed in claim **11**, wherein the plurality of display modes includes a locked mode in which the second display layer is locked in an invisible state.

15. A graphical user interface as claimed in claim **11**, wherein the plurality of display modes includes a timed icon mode in which the second display layer is normally displayed as a visible icon and is in a visible state when the position indicator is coincident with the icon, then returns to an invisible state after a predetermined time.

16. A graphical display system for an application program, comprising:

a user interface manager for storing application oriented objects associated with said application program, hierarchical display lists of said objects, and raster images of said objects, said objects including representations each defined by an overall display layer and position, individually layered and positioned object components, and symbolic references to graphical constructs for each said component, a plurality of logical states defined by keys linked to representation changes and a plurality of behaviors defined by actions linked to symbolic object events;

a renderer module responsive to a presentation manager for maintaining said raster images according to the contents of said display lists and copying one or more of said raster images to hardware video memory for display on a display device; and

a presentation manager responsive to a motion event or user input for invoking said user interface module and said renderer.

17. A graphical display system as defined in claim **16**, said presentation manager being operable to insert, modify, and delete graphical data in said display lists to reflect the representation properties of said application-oriented objects; maintain the hierarchical ordering of said display lists to reflect layering properties of said application-oriented objects; maintain a mapping from all graphical data in said display lists to application-oriented objects with which data is associated; translate operating system input events to symbolic object events on an object-by-object basis; and invoke user interface manager actions as required to implement predetermined behaviors of said objects.

18. A graphical display system as defined in claim **16**, said display lists containing graphical data describing the appearance of the display and including graphical data consisting of vector-based graphical primitives and symbolic graphical attributes including color, drawing style, and/or text characteristics, a hierarchical structure in which any display list may contain other display lists and graphical data and in which the display lists, their contained display lists, and graphical data are sorted from a lowest underlying layer to highest overlying layer relative to a containing display list.

19. A graphical display system as defined in claim 16, said renderer being operable to traverse the display list hierarchy in either ascending or descending sort order and perform vector-by-vector conversion of graphical primitives to raster images constrained by an image extent or distance from a specific image pixel; determine a set of rendered pixels for each primitive based on primitive vectors and a translation of the symbolic graphic attributes for drawing styles and/or text characteristics; determine a value to be applied against a raster image for each rendered pixel based on a translation of the symbolic graphic attributes for color; determine a logical function for each rendered pixel specifying the manner in which a pixel value is to be applied against a raster image based on a translation of symbolic attributes for color, drawing styles, and/or text characteristics such that for transparent pixels, said logical function leaving the corresponding pixel in the raster image unchanged; for opaque pixels, said logical function replacing the corresponding pixel in the raster image and other logical functions provide different visual effects; updating one or more raster images from the rendered pixel set, values, and functions determined from a primitive; and copying some or all of one or more raster images to the hardware video memory.

20. A graphical display system as defined in claim 16, said renderer being operable to either map said raster images directly to video memory to graphics display hardware or block copy said raster images directly to the video memory, said raster images being comprising an array of pixel values arranged exactly as required for video display, said renderer being further operable to render overlying objects after rendering underlying objects so that pixels rendered for the overlying objects modify or replace those of underlying objects.

21. A graphical display system as defined in claim 17, said display lists containing graphical data describing the appearance of the display and including graphical data consisting of vector-based graphical primitives and symbolic graphical attributes including color, drawing style, and/or text characteristics, a hierarchical structure in which any display list may contain other display lists and graphical data and in which the display lists, their contained display lists, and graphical data are sorted from a lowest underlying layer to highest overlying layer relative to a containing display list.

22. A graphical display system as defined in claim **21**, said renderer being operable to traverse the display list hierarchy in either ascending or descending sort order and perform vector-by-vector conversion of graphical primitives to raster images constrained by an image extent or distance from a specific image pixel; determine a set of rendered pixels for each primitive based on primitive vectors and a translation of the symbolic graphic attributes for drawing styles and/or text characteristics; determine a value to be applied against a raster image for each rendered pixel based on a translation of the symbolic graphic attributes for color; determine a logical function for each rendered pixel specifying the manner in which a pixel value is to be applied against a raster image based on a translation of symbolic attributes for color, drawing styles, and/or text characteristics such that for transparent pixels, said logical function leaving the corresponding pixel in the raster image unchanged; for opaque pixels, said logical function replacing the corresponding pixel in the raster image and other logical functions provide different visual effects; updating one or more raster images from the rendered pixel set, values, and functions determined from a primitive; and copying some or all of one or more raster images to the hardware video memory.

23. A graphical display system as defined in claim **22**, said renderer being operable to either map said raster images directly to video memory to graphics display hardware or block copy said raster images directly to the video memory, said raster images being comprising an array of pixel values arranged exactly as required for video display, said renderer being further operable to render overlying objects after rendering underlying objects so that pixels rendered for the overlying objects modify or replace those of underlying objects.

24. A method of managing a graphical user interface associated with an application program operable on a computing device having a visual display device, a user input device for controlling a pointer on said display device, said application having a plurality of windows including a primary background window and one or more secondary windows overlying said background window, each said secondary window being configurable in normal, locked, timed and timed icon modes and in invisible, opaque, and iconified states, comprising the steps of:

a) storing predetermined data with respect to each said window;
b) storing hierarchical display lists of said windows;
c) storing raster images of said windows;
d) storing the identity of an window underlying said pointer as a current window;
e) responding to movement of said pointer over said windows by:
    i. comparing the identity of the window immediately underlying said pointer against the identity of a previous window;
    ii. determining the mode of the previous window and
        a. placing said previous window in an invisible state when said previous window is in a normal mode; and
        b. initiating an invisible time out when said previous window is in a timed or timed icon mode;
    iii. removing an invisible state from said current window;
    iv. storing the identity of said current window as said previous window; and
    v. removing any pending time outs respecting said current window.

25. A method as defined in claim **24**, said step of storing predetermined data with respect to each said window including storing representations each defined by an overall display layer and position, individually layered and positioned object components, and symbolic references to graphical constructs for each said component, and the mode and state of said windows.

26. A method as defined in claim **24**, said step of initiating an invisible time out further including setting said previous window in an said invisible when a time out event occurs when said previous window was in a timed mode and setting said previous window in an said iconized state when a time out event occurs when said previous window was in a timed icon mode.

27. A method as defined in claim **24**, wherein in said normal mode the contents of the window are exposed when said pointer moves into the area of said window, said window contents being exposed either by displaying the window displayed on an opaque background to enhances legibility of the window contents, or by displaying said window on a transparent background to enable the contents of background windows to be visible underneath the invisible window.

28. A method as defined in claim **24**, wherein, in said "timed" mode, the contents of the window are exposed a predetermined period of time when said pointer moves into the area of said window, said window contents being exposed either by displaying the window displayed on an opaque background to enhances legibility of the window contents, or by displaying said window on a transparent background to enable the contents of background windows to be visible underneath the invisible window, and at the end of said predetermined period of time, returning the window to said invisible state.

29. A method as defined in claim **24**, wherein, in said "timed icon" mode, the contents of the window are exposed a predetermined period of time when said pointer moves into the area of said window, said window contents being exposed either by displaying the window displayed on an opaque background to enhances legibility of the window contents, or by displaying said window on a transparent background to enable the contents of background windows to be visible underneath the invisible window, and at the end of said predetermined period of time, reducing the window to said icon state.

30. A method as defined in claim **24**, further including the step of updating windows disposed beneath transparent windows including the steps of:

a) setting a current display list to a top level display list;

b) for each item in said display list:

i. if said item is not an embedded display list, determining the rendered pixel set, pixel values and logical functions associated with said item, and updating raster images from the rendered pixels associated with said item;

ii. if said item is an embedded display list, repeating steps a) and b) with respect to said embedded display list;

and

c) when all items in all display lists have been processed, copying updated images to video memory.

31. A method as defined in claim **24**, further including the step of implementing the behavior of objects which are invisible, in response to movement of a tracking device,

a) determining the pixel coordinates of said tracking device from a tracking event;

b) determining which primitives in the display lists, searched from uppermost to lowermost layers, contain said pixel coordinates in their rendered pixel set;

c) if said lists contain at least one primitive, for each such primitive:

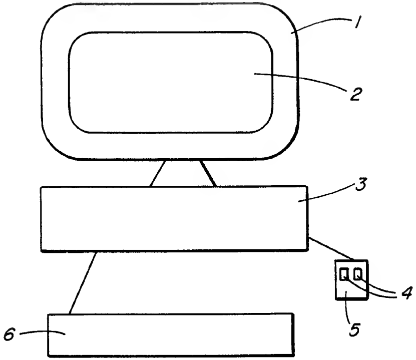
i. accessing the application-oriented object associated with said at least one primitive;

ii. invoking the actions linked to the object event according to a predetermined behavior definition;

iii. if said event allows event propagation, propagating said event to the next underlying object;

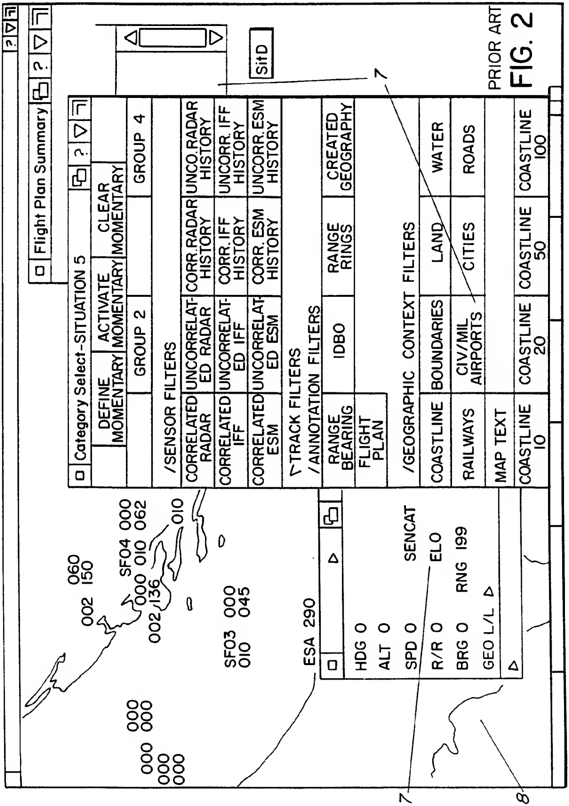
d) If said event is to be propagated or if said event cannot be mapped to a symbolic object event, deleting the primitive from the list.

32. A computer readable memory or article of manufacture for storing the instructions or statements for use in the execution of the method of claim 24.



PRIOR ART

FIG. 1



PRIOR ART
FIG. 2

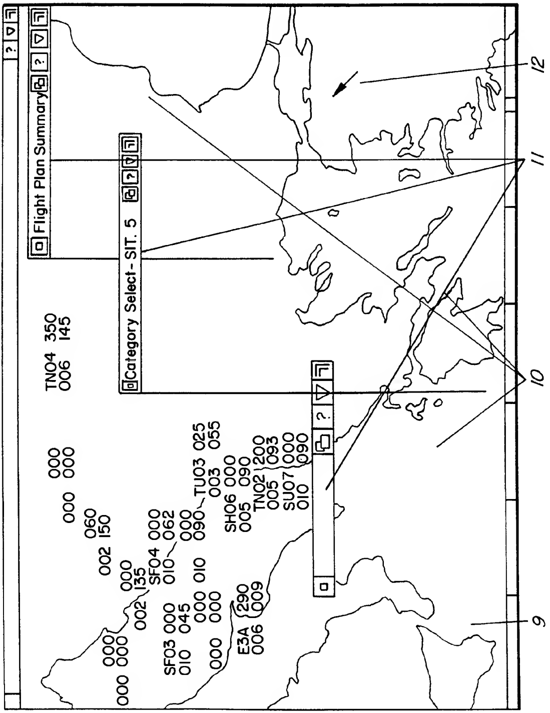


FIG. 3

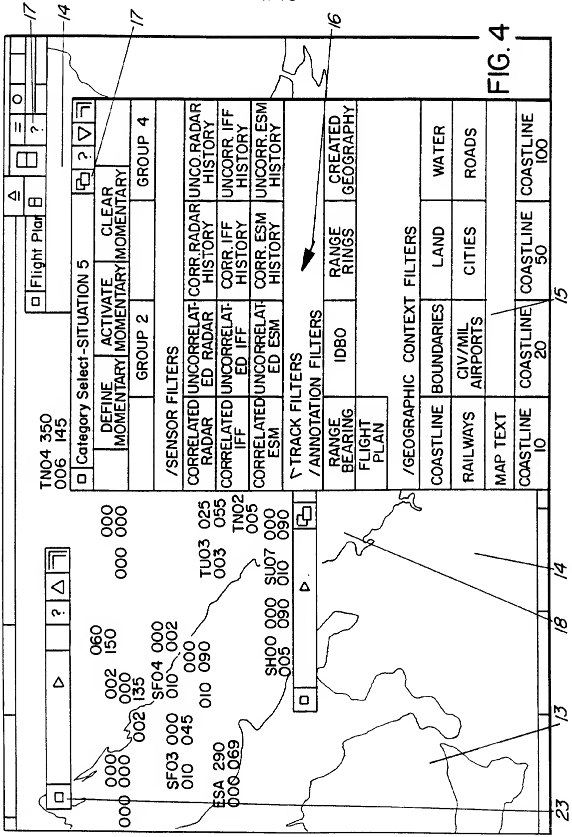


FIG. 4

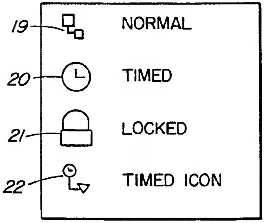


FIG. 5

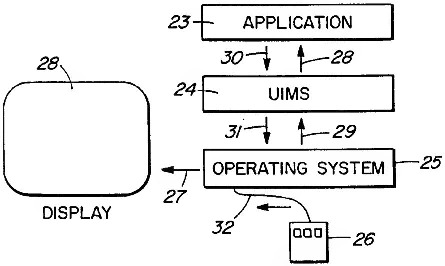


FIG. 6

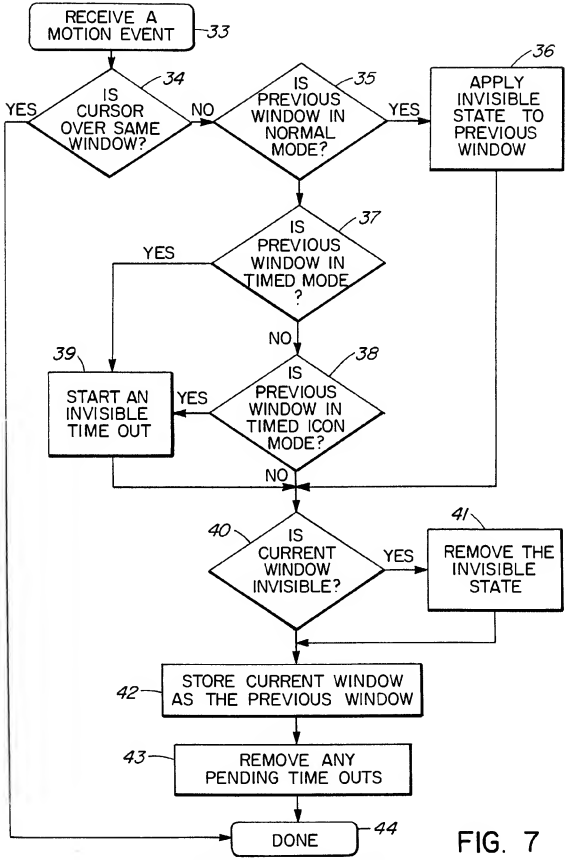


FIG. 7

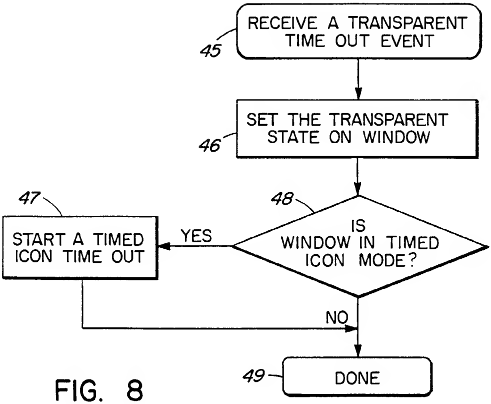


FIG. 8

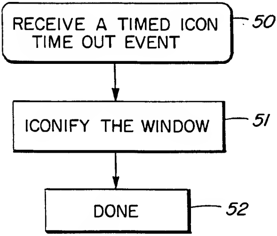


FIG. 9

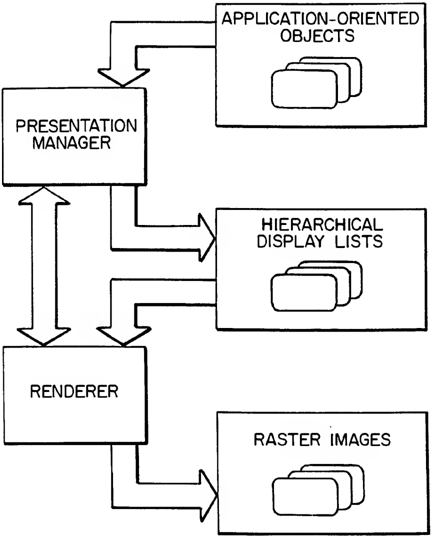


FIG. 10

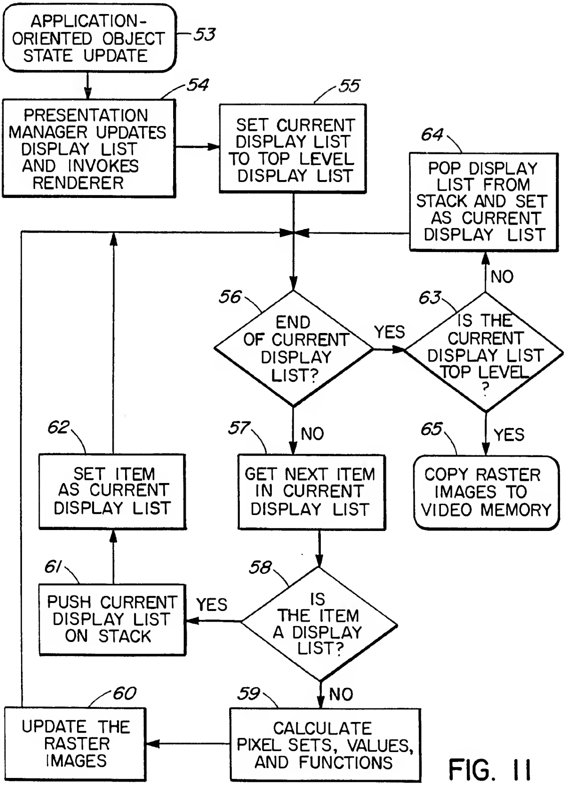
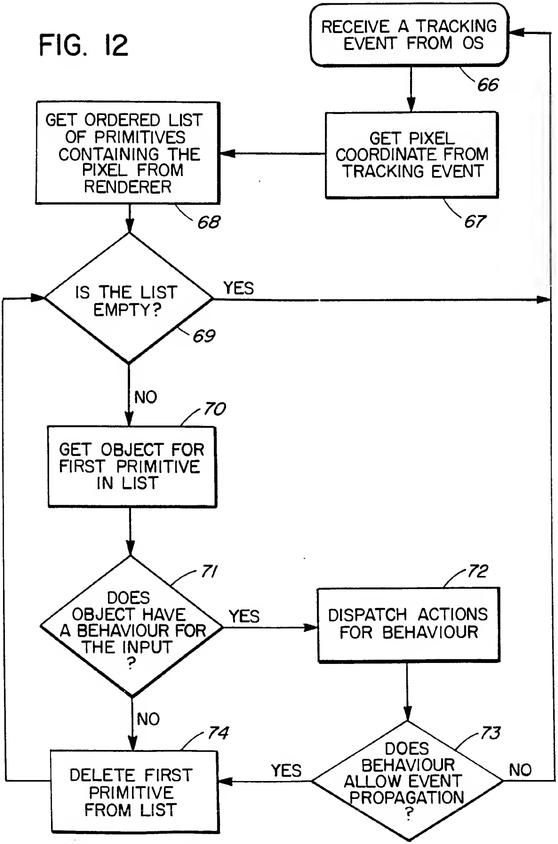


FIG. 11

FIG. 12



INTERNATIONAL SEARCH REPORT

International Application No
PCT/CA 98/00125

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F3/033 G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y A	EP 0 439 873 A (IBM ;QUEEN MARY & WESTFIELD COLLEGE (GB)) 7 August 1991 see column 2, line 40 - line 50 see column 3, line 10 - line 33 see column 4, line 15 - line 57; figures 3,4	1-4,6-9 16,24,27
Y A	EP 0 349 459 A (IBM) 3 January 1990 see the whole document	1-4,6-9 11,12
A	PATENT ABSTRACTS OF JAPAN vol. 096, no. 011, 29 November 1996 & JP 08 171476 A (CANON INC), 2 July 1996 see abstract	3,8,13, 15,24, 26,28,29

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
29 July 1998	12/08/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Fonderson, A

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/CA 98/00125

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	"AUTOMATIC WINDOW MANAGEMENT MODE" IBM TECHNICAL DISCLOSURE BULLETIN, vol. 35, no. 4B, 1 September 1992, pages 331-332, XP000313970 see the whole document	1-4,6-9, 11,12,15
A	EP 0 747 805 A (SONY ELECTRONICS INC A DELAWAR) 11 December 1996 see column 1, line 1 - column 3, line 45 see column 19, line 26 - line 52	1,6,11, 24,27
A	IBM: "OS/2 2.0 Presentation Manager Graphics Programming Guide" March 1992 , QUE , USA XP002073056 see page 12-1, line 1 - page 12-8, last line; figures 12-2,12-3 see page 13-1, line 1 - page 13-12, last line; figures 13-2,13-3 see figures 13-4	16-25, 30,31
A	US 5 420 968 A (JOHRI ABHAI) 30 May 1995 see column 1, line 35 - line 50; figure 5 see column 2, line 35 - column 3, line 36	11,14
A	US 5 590 265 A (NAKAZAWA TOSHIYUKI) 31 December 1996 see abstract; figure 1	1-32

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int tional Application No
PCT/CA 98/00125

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0439873 A	07-08-1991	DE 69022210 D	12-10-1995
		DE 69022210 T	02-05-1996
		JP 2723208 B	09-03-1998
		JP 3245254 A	31-10-1991
		US 5293470 A	08-03-1994
EP 0349459 A	03-01-1990	US 5046001 A	03-09-1991
		CA 1315894 A	06-04-1993
		DE 68923492 D	24-08-1995
		DE 68923492 T	07-03-1996
		JP 2014771 C	02-02-1996
		JP 2025960 A	29-01-1990
		JP 7054506 B	07-06-1995
EP 0747805 A	11-12-1996	US 5675755 A	07-10-1997
		CA 2177801 A	08-12-1996
		JP 9146751 A	06-06-1997
US 5420968 A	30-05-1995	NONE	
US 5590265 A	31-12-1996	JP 6051738 A	25-02-1994